Achieving PCI Transparency With HyperTransport™

Mark Hummel
Silicon Architect
API NetWorks, Inc.

Platform Conference
Direction • Design • Perspective • Analysis

Objectives

- No driver changes for PCI devices
- Minimal BIOS changes
- PCI and HyperTransport™ devices fully interoperate
- Common programming model for PCI and HyperTransport™

Compatibilities

- PCI bus bridging
- Producer/consumer ordering
- Configuration/IO/memory space
- Device/bridge headers
- Device bandwidth allocation
- Error handling

PCI Bus Bridging

- Posted request support
- Deadlock free peer-to-peer operation
- Read prefetching support
- Single unified bus/device id Enumeration

Posted Request Support

- Responseless write operation
- Uses less link BW
- Stateless in bridges

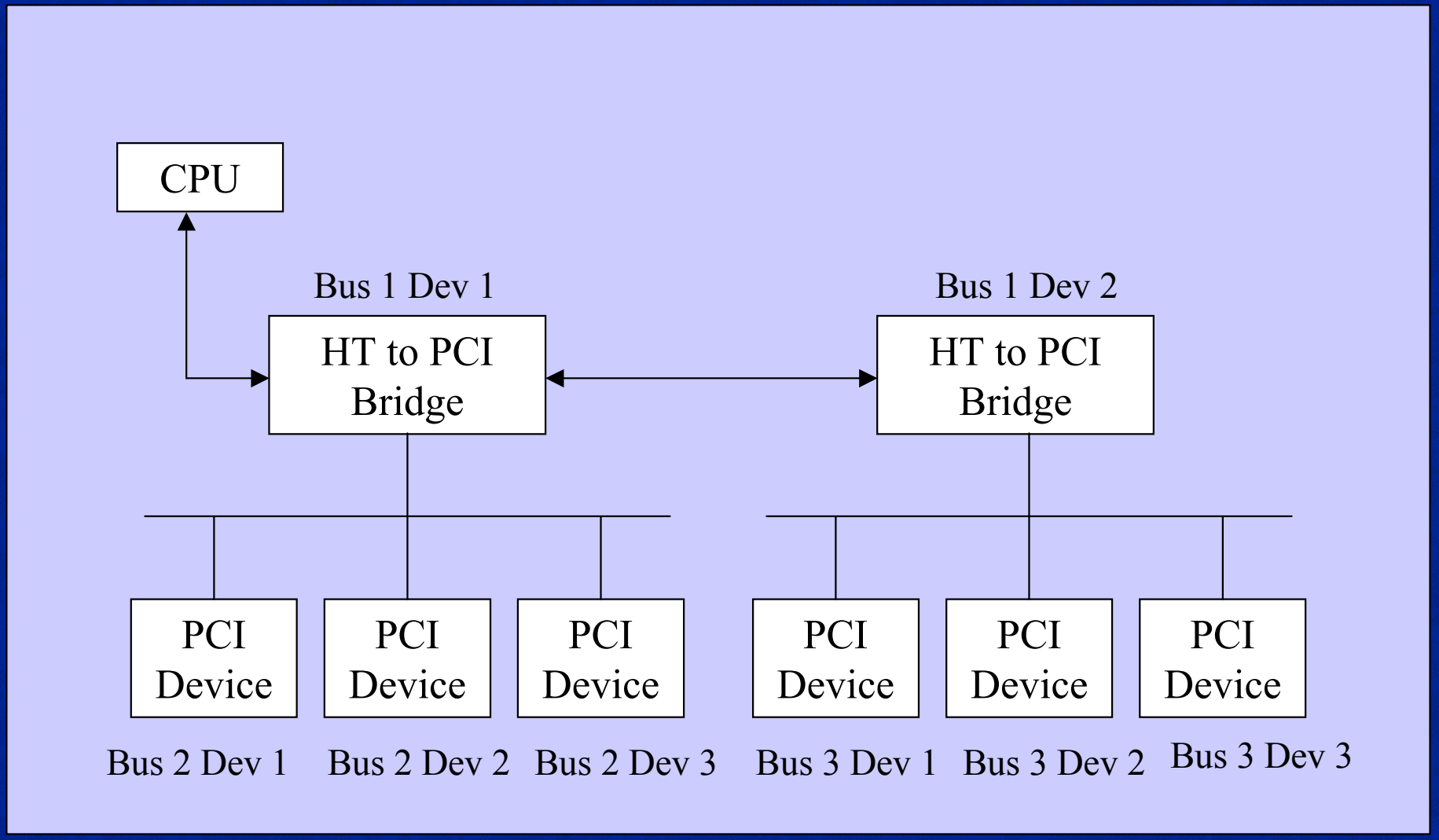
PCI Bus Bridging

- **Three virtual channels**
  - Posted
  - Non-posted
  - Response
- **PCI request mapping**
  - Memory writes => Posted channel
  - IO/Configuration writes => Non-posted channel
  - Reads => Non-posted channel
  - Responses => Response channel

Read Prefetching Support

- PCI delayed reads must be prefetched to be efficient for burst transactions
- Stream of HyperTransport™ Reads may be issued with a common SeqId to ensure proper read ordering

Bus/Device ID Enumeration



Producer/Consumer Ordering

- **Supports PCI ordering semantics**

Flag = 0 Data = 0

Producer	Consumer
Wr Data <= 1	f = Rd Flag
Wr Flag <= 1	d = Rd Data

Must satisfy: !(d == 0 && f == 1)

Configuration/IO/Memory Space

- **Separate configuration, IO and memory address ranges**
- **Configuration addresses controlled by bus number registers**
- **IO/prefetchable/memory spaces specified like PCI**
  - Bridges use range registers
  - Devices use BARS
- **Subtractive decode support**

Device/Bridge Headers

- **Device header is subset of PCI device header**
- **Bridge header is a subset of PCI bridge header**
- **HyperTransport™ specific extensions accessed via capability block**
  - Set by initialization code (BIOS)
  - Device drivers may require enhancement to fully utilize extensions

Device Header

Device ID		Vendor ID	
Status		Command	
Class Code			Revision ID
BIST	Header Type	Latency Timer	Cache Line Size
Base Address Register			
Cardbus CIS Pointer			
Subsystem ID		Subsystem Vendor ID	
Expansion ROM Base Address			
Reserved			Capabilities Pointer
Reserved			
Min_Lat	Min_Gnt	Interrupt Pin	Interrupt Line

Bridge Header

Device ID		Vendor ID	
Status		Command	
Class Code			Revision ID
BIST	Header Type	Primary Latency Timer	Cache Line Size
Base Address Register 0			
Base Address Register 1			
Secondary Latency Timer	Subordinate Bus Number	Secondary Bus Number	Primary Bus Number
Secondary Status		I/O Limit	I/O Base
Memory Limit		Memory Base	
Prefetchable Memory Limit		Prefetchable Memory Base	
Prefetchable Base Upper 32 Bits			
Prefetchable Limit Upper 32 Bits			
I/O Limit Upper 16 Bits		I/O Base Upper 16 Bits	
Reserved			Capabilities Pointer
Expansion ROM Base Address			
Bridge Control		Interrupt Pin	Interrupt Line

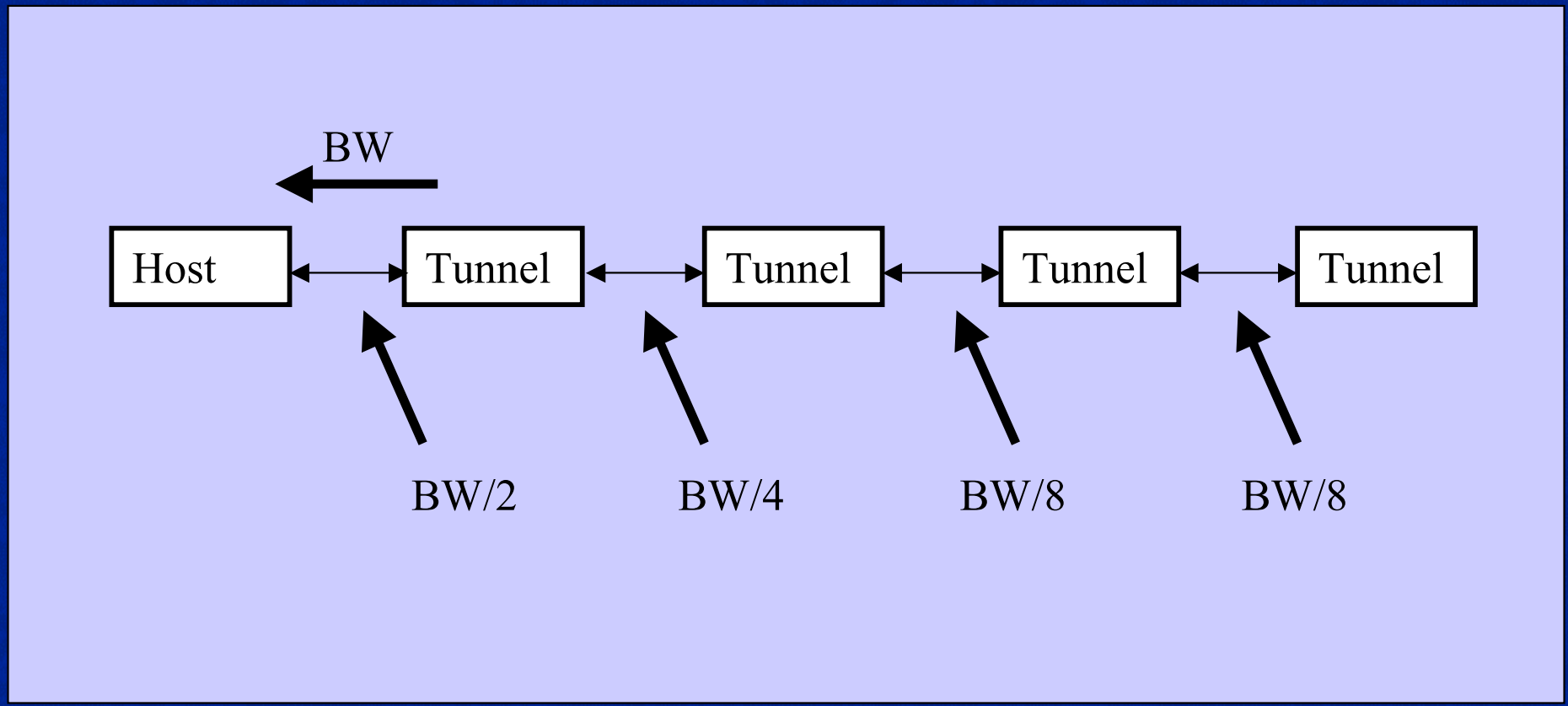
Capability Block

Command	Capabilities Pointer		Capability ID
Link Config 0	Link Control 0		
Link Config 1	Link Control 1		
LinkFreqCap0	Link Error 0	Link Freq 0	HT Rev ID
LinkFreqCap1	Link Error 1	Link Freq 1	Feature
Error Handling	Enumeration Scratchpad		
Reserved	Mem Limit Upper		Mem Base Upper

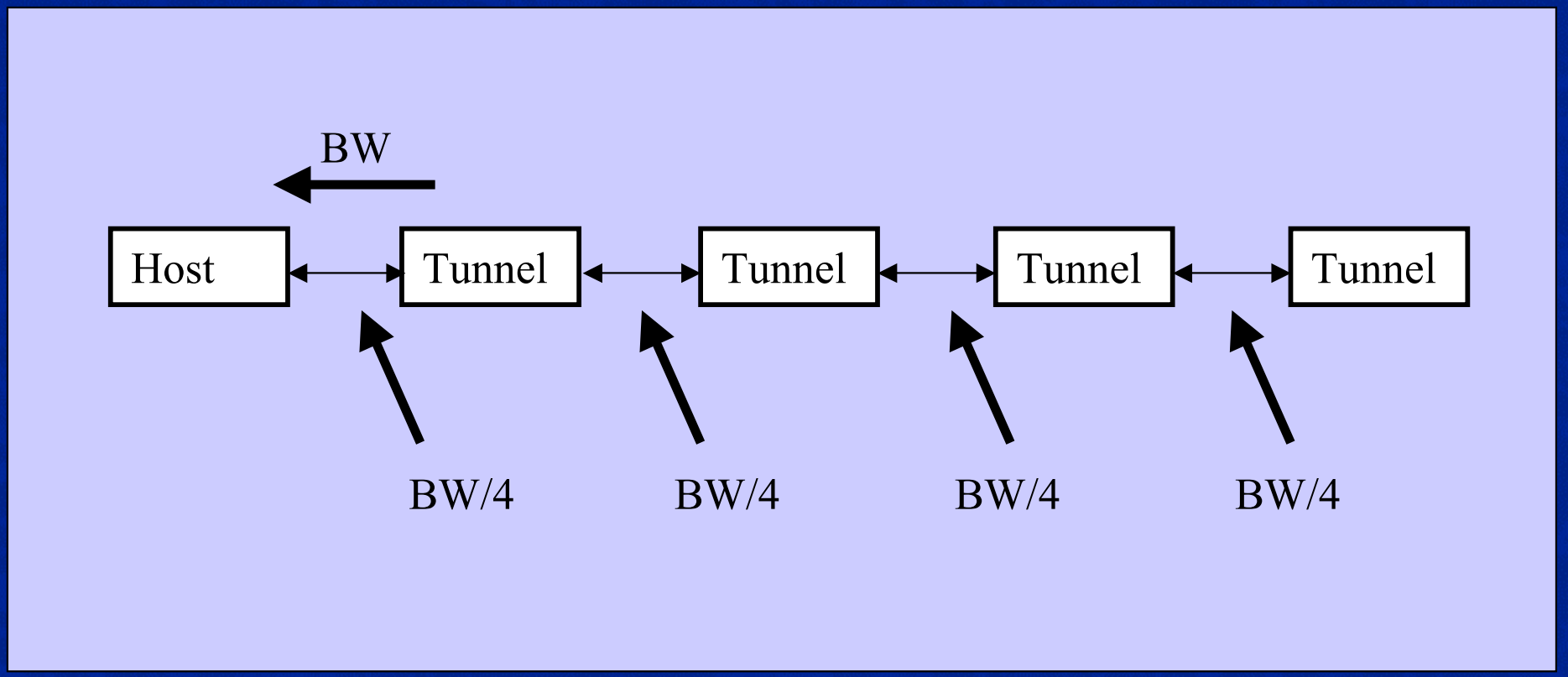
Bandwidth Allocation

- Chain of tunnels without insertion rate control results in unfair bandwidth allocation
- Dynamic rate insertion approximates round robin bus arbitration

Without Rate Control



Dynamic Rate Control



Error Handling

- **Master Abort**
  - Response with NXA=1,ERROR=1
- **Target Abort**
  - Response with NXA=0,ERROR=1
- **SERR**
  - Signaled using SYNC FLOOD
- **Bridges rectify Master Aborts based on Master-Abort Mode CSR bit**
  - Master-Abort Mode=0 => Returns all 1's Data,
  - Master-Abort Mode=1 => Convert to Target Abort

Differences

- Initialization
- Interrupts

Initialization

- Link initialization unique to HyperTransport™
- Link initialization can be run first, followed by PCI discovery/initialization software.

Link Initialization

- **Device configuration done by initialization code (BIOS)**
  - Device discovery
  - Unit ID assignment
  - Frequency and width negotiation
  - Interrupt Configuration

Interrupts

- Interrupts are transported via packets, not wires
- Interrupt packets carry opaque info, generated by devices, interpreted by hosts
- Built-in support for edge and level sensitive interrupts

Interrupts (Cont.)

- Interrupt mappings setup by initialization code (BIOS)
- APIC emulation support for X86 platforms

Conclusion

- PCI programming model
- HyperTransport™ and PCI fully interoperate
- Interrupts and initialization are main differences from PCI